Simultaneous drone localisation and wind turbine model fitting during autonomous surface inspection

Oliver Moolan-Feroze[1], Konstantinos Karachalios[2], Dimitrios N. Nikolaidis[2], and Andrew Calway[1]

***Abstract*— We present a method for simultaneous localisation and wind turbine model fitting for a drone performing an automated surface inspection. We use a skeletal parameterisation of the turbine that can be easily integrated into a non-linear least squares optimiser, combined with a pose graph representation of the drone's 3-D trajectory, allowing us to optimise both sets of parameters simultaneously. Given images from an onboard camera, we use a CNN to infer projections of the skeletal model, enabling correspondence constraints to be established through a cost function. This is then coupled with GPS/IMU measurements taken at key frames in the graph to allow successive optimisation as the drone navigates around the turbine. We present two variants of the cost function, one based on traditional 2D point correspondences and the other on direct image interpolation within the inferred projections. Results from experiments on simulated and real-world data show that simultaneous optimisation provides improvements to localisation over only optimising the pose and that combined use of both cost functions proves most effective.**

I. INTRODUCTION

Over the operational lifetime of a wind turbine, it will likely incur a wide range of structural damage [1] due to adverse weather and events such a lightning strikes and bird collisions. For operational efficiency, this damage needs to be detected and fixed as soon as possible [2]. This currently relies on manual inspections using climbing equipment, or the use of telephoto lenses from the ground. From the danger incurred during manual inspection, to the inability to get full coverage of the turbine surface using ground based cameras, it is evident that both methods have downsides.

One way of addressing the problem is the use of unmanned autonomous vehicles (UAVs) – or drones – to perform inspections [3]. This provides a number of benefits. It is safe, as no human personnel are required to climb the turbine. Also, as it is able to navigate freely around the turbine, it is able to obtain a complete view of the surface, including from above. Another significant benefit is the abilitiy to perform consistent and repeatable inspections.

Key to the drone's inspection performance is its ability to navigate around the turbine. To enable this, it needs an accurate estimate of its 3-D pose w.r.t the turbine, as well as an accurate estimate of the turbine's structure and configuration. Having these allows faster and safer navigation, more consistent inspections, and provides important information needed for post-processing operations such a image stitching and automatic default detection. Although the drone has access to a global positioning system (GPS) and inertial measurement units (IMUs) to aid in localisation, it has been shown that incorporating image measurements can improve accuracy significantly [4], [5].

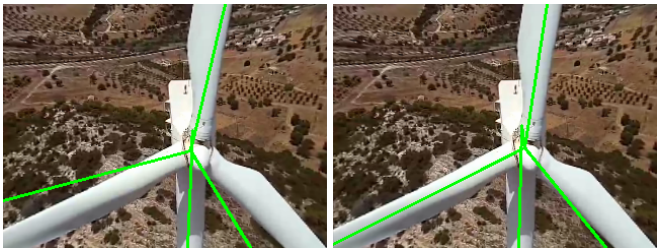

Fig. 1. Figure showing the benefit of simultaneous model fitting and pose estimation. The green lines show the reprojection of the wind turbine using the optimised model parameters and the pose estimates. Left) Turbine reprojection when only the pose is optimised. Right) Turbine reprojection when both pose and model parameters are optimised.

We present a method that is able to simultaneously optimise the drone's pose, as well as fit a model of the turbine during inspection. These two processes are complementary, in that the estimation of the turbine model is needed for accurate localisation, and localisation is required for fitting the model. The joint optimisation is achieved using a skeletal parameterisation of the turbine structure, accompanied with a set of functions that are able to instantiate the model. These functions, along with a pose graph representation of the drone's localisation are integrated into a non-linear least squares optimiser. We also present a novel cost function based on direct image interpolation that is better suited to this application than the typical cost based on 2D corresponding points. The work builds on our localisation method previously described in [5].

There are two main contributions. The first is the integration of the turbine model fitting process into the non-linear least squares optimiser. The skeletal representation and set of functions were carefully chosen so that they could be easily integrated into the optimiser as well as providing a model that fully describes the main structures of the turbine and is sufficiently general so as to be used with a range of turbine configurations and types. The second contribution is the use of an interpolation-based cost measure, which proves to be effective for optimising over line structures in the skeletal model and complements the point based approach described in [5].

[1]Oliver Moolan-Feroze and Andrew Calway are with the Department of Computer Science, University of Bristol, 75 Woodland Road, Bristol, BS8 1UB, United Kingdom oliver.moolan-feroze@bristol.ac.uk, andrew.calway@bristol.ac.uk

[2]Konstantinos Karachalios and Dimitrios N. Nikolaidis are with Perceptual Robotics, 5 Hope Road, Bristol, UK, BS3 3NZ, United Kingdom kostas@perceptual-robotics.com, dimitris@perceptual-robotics.com

In Section II we highlight some of the relevant literature related to this work. In Section III, we give a detailed description of the proposed method. This includes the parameterisation of the model and accompanying functions in Section III-A, the production of the image measurements using a CNN in Section III-B, and the joint optimisation of the pose and model parameters in Section III-C. In Section IV, we show our evaluation of the method. This was done using both simulated data in Section IV-A as well as real data in Section IV-B. In Section V we present some conclusions as well as potential avenues for future work.

II. PREVIOUS WORK

Likely due to the relative novelty of the application and the difficulty in acquiring data, there is little published research on the autonomous inspection of wind turbines using drones. Moreover, the works that tackle this problem have a number of important omissions. In [6], a method is presented for the prediction of the configuration of the wind turbine during an initial approach of the drone. They use a combination of the Hough Transform to extract edges and estimate the state of the turbine and a Kalman Filter to keep track of the drone's position. One problem with this work is that it only addresses the initial approach of the drone, and does not attempt to estimate the parameters during the remaining – and most important – part of the inspection. In [7], the authors suggest the use of LiDAR sensors to both estimate the state of the turbine and the position of the drone. The turbine model is represented as a voxel grid which is filled in using a Bayesian update scheme during the inspection process. The grid is then used for path planning. One drawback of this work is that it is only applied in simulation. Furthermore, the authors make no attempt at the simultaneous localisation of the drone.

Although not directly related to wind turbine inspection, there are a number of methods that attempt simultaneous localisation and model fitting. These works are mainly focussed on the use of prior knowledge to improve the performance of simultaneous localisation and mapping (SLAM). In [8], a method using RGBD data is proposed which makes use of a partial scene model that can constrain the optimiser, improving performance when using noisy and inaccurate depth data. Similarly, the work by Loesch et. al. [9], describes the integration of known objects into a SLAM process. Using an edglet-based feature description of the object, the method is able to constraint the SLAM process and reduce absolute drift in the localisation estimates.

Research into articulated object pose detection is relevant to the model fitting part of our method. Pauwels et. al. [10] propose an Iterative Closest Point (ICP) method that is tailored to articulated models. This is achieved using a set of constraints on the objects which restrict the possible movements between object parts based on the current view. Constraints are also used in [11], in the form of a kinematic chain. This allows the authors to fully localise multi part objects by estimating only the parameters of the chain rather than the full 6DoF poses of the parts. To aid in this they make use of a random forest to predict object part probabilities and object coordinates. Katz et. al. [12] also make use of a kinematic chain, extracting and tracking visual features as a robot is interacting with articulated objects. Using clustering, tracked features are grouped into sets that represent different parts of the articulated object. From these, trajectories are established which define the types of constraints needed by different object parts.

The work presented here is an extension of the localisation method described in [5], which optimises the pose of the drone using a fixed skeletal turbine model. It used a convolutional neural network (CNN) to infer the projection of the model in images captured onboard, which are then integrated into a pose graph optimisation framework. We follow a similar approach in this work, except that we simultaneously estimate the turbine model parameters, producing improved localisation results and providing greater flexibility in the use of the method. In the following, we omit some of the detail regarding the CNN due to space constraints and refer readers to [5] for a fuller description.

III. METHOD

The joint turbine model fitting and pose estimation process is performed throughout the duration of an inspection flight. We model the problem as a non-linear least squares optimisation over the set of model parameters and pose parameters. In Section III-A we detail the parameterisation of the wind turbine model, and describe the functions that instantiate the model. In Section III-B we describe our CNN-based model projection estimation method. The projection estimates allow us to easily find correspondences between the model and image features. In Section III-C we describe the optimisation process, detailing the cost function and how it is optimised.

A. Turbine Model Parameterisation

There are two aims determining the selection of the parameterisation of the wind turbine model. The first is that it should be as minimal as possible to simplify the optimisation process. The second is that the parameterisation must be complete, in that it fully describes the structure of the turbine. This enables the drone's navigation system to make path planning decisions based on it. The parameterisation we have chosen is based on the following values:

- The 2D location of the turbine's base on the xy plane: $\mathbf{c} \in \mathbb{R}^2$
- The height of the turbine's tower: h
- The heading of the turbine relative to the drone's coordinate system: ω
- The length of the turbine nacelle which joins the top of the turbine tower with the blade centre: r
- The rotation angle of the turbine blades: ϕ
- The length of the turbine blades: b

Using the parameterisation above $\theta = \{\mathbf{c}, h, \omega, r, \phi, b\}$, a number of different turbine models can be instantiated. During the optimisation process we make use an instantiation based on points $\mathcal{P}$ and an instantiation based on lines $\mathcal{L}$. It should be noted that during an inspection, the blades will

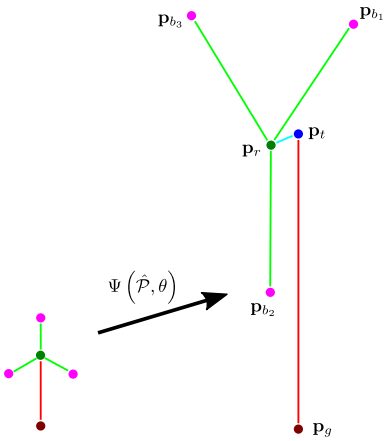

Fig. 2. Image showing the model instantiation. $\hat{\mathcal{P}}$ is to the left. The instantiated model $\mathcal{P}$ is to the right.

be fixed in place. This means that the rotation angle of the blades is static during the flight.

1) Turbine Point Model: The point model consists of a set of six points $\mathcal{P} = \{\mathbf{p}_g, \mathbf{p}_t, \mathbf{p}_r, \mathbf{p}_{b_1}, \mathbf{p}_{b_2}, \mathbf{p}_{b_3}\}$ with $\mathbf{p} \in \mathbb{R}^3$. Each point represents a key location on the wind turbine structure. These are the turbine base $\mathbf{p}_g$, the top of the turbine tower $\mathbf{p}_t$, the centre of the turbine blades $\mathbf{p}_r$, and the three tips of the blades $\mathbf{p}_{b_i}$ where $i \in \{1, 2, 3\}$. To instantiate a point model $\mathcal{P}$ given a set of parameters θ, we make use of a "default" model $\hat{\mathcal{P}}$. This "default" model consists of the full set of model points $\hat{\mathbf{p}}$, initialised to the following values

$$\hat{\mathcal{P}} = \begin{cases} \hat{\mathbf{p}}_g & = & (0,0,0) \\ \hat{\mathbf{p}}_t & = & (0,0,1) \\ \hat{\mathbf{p}}_r & = & (1,0,1) \\ \hat{\mathbf{p}}_{b_i} & = & (1,0,2) \end{cases} \tag{1}$$

Note that in the "default" model, the tower is of unit height, the blades are of unit length, and the heading is oriented along the x-axis. We now define the set of functions $\Psi = \{\boldsymbol{\psi}_g, \boldsymbol{\psi}_t, \boldsymbol{\psi}_r, \boldsymbol{\psi}_{b_i}\}$ that take the points defined in $\hat{\mathcal{P}}$ and the turbine parameters θ to create an instance of the turbine point model model $\mathcal{P}$

$$\mathcal{P} = \begin{cases} \mathbf{p}_g & = & \boldsymbol{\psi}_g\left(\hat{\mathbf{p}}_g, \theta\right) \\ \mathbf{p}_t & = & \boldsymbol{\psi}_t\left(\hat{\mathbf{p}}_t, \theta\right) \\ \mathbf{p}_r & = & \boldsymbol{\psi}_r\left(\hat{\mathbf{p}}_r, \theta\right) \\ \mathbf{p}_{b_i} & = & \boldsymbol{\psi}_{b_i}\left(\hat{\mathbf{p}}_{b_i}, \theta\right) \end{cases} \tag{2}$$

The function ψ_g that computes the location of the point at the base of the turbine $\mathbf{p}_g$ is defined as

$$\boldsymbol{\psi}_g(\hat{\mathbf{p}}_g, \theta) = \mathbf{p}_\mathbf{c} + \hat{\mathbf{p}}_g \tag{3}$$

where $\mathbf{p}_\mathbf{c} = (\mathbf{c}, 0)$ maps $\mathbf{c}$ into $\mathbb{R}^3$. To compute the location of the point at the top of the turbine tower $\mathbf{p}_t$, we use

$$\boldsymbol{\psi}_t(\hat{\mathbf{p}}_t, \theta) = \mathbf{p}_\mathbf{c} + \mathbf{M}_h\hat{\mathbf{p}}_t \ , \tag{4}$$

where $\mathbf{M}_h = \text{diag}(1, 1, h)$ scales the input point along the z-axis. The point at the centre of the blades $\mathbf{p}_r$ is computed using

$$\boldsymbol{\psi}_r(\hat{\mathbf{p}}_r, \theta) = \mathbf{p}_\mathbf{c} + \mathbf{R}_\omega\mathbf{M}_r\mathbf{M}_h\hat{\mathbf{p}}_r \ , \tag{5}$$

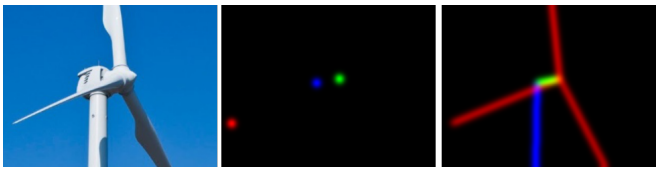

Fig. 3. Example outputs from the CNN. Left) Input image. Centre) The predicted projection of $\mathcal{P}$, with different parts $\left\{\mathbf{P}^{\mathcal{P}}_{n,g}, \mathbf{P}^{\mathcal{P}}_{n,t}, \mathbf{P}^{\mathcal{P}}_{n,r}, \mathbf{P}^{\mathcal{P}}_{n,b}\right\}$ coloured differently. Right) The predicted projection of $\mathcal{L}$ with the different parts $\left\{\mathbf{P}^{\mathcal{L}}_{n,t}, \mathbf{P}^{\mathcal{L}}_{n,r}, \mathbf{P}^{\mathcal{L}}_{n,b}\right\}$ coloured differently.

where $\mathbf{M}_r = \text{diag}(r, 1, 1)$ scales the input point along the x-axis, and $\mathbf{R}_\omega \in \mathbb{R}^{3x3}$ is the rotation matrix that rotates the scaled points around the z-axis by an angle of ω. The turbine blade tips $\mathbf{p}_{b_i}$ are computed in the following manner

$$\boldsymbol{\psi}_{b_i}(\hat{\mathbf{p}}_{b_i}, \theta) = \mathbf{p}_\mathbf{c} + \mathbf{R}_\omega\mathbf{M}_r\left(\mathbf{R}_{\phi_i}\mathbf{M}_b\left(\hat{\mathbf{p}}_{b_i} - \hat{\mathbf{p}}_t\right) + \mathbf{M}_h\hat{\mathbf{p}}_t\right), \tag{6}$$

where $\mathbf{M}_b = diag(1, 1, b)$ scales the length of the blades along the z-axis. The rotation $\mathbf{R}_{\phi_i}$ is the matrix that rotates the blades around the x-axis by ϕ_i. For the three blade tips, the values for each ϕ_i are $\left(\phi, \phi + \frac{2}{3}\pi, \phi + \frac{4}{3}\pi\right)$. These values produce three blade tips, rotated 120° from each other. A diagram showing an instantiated point model $\mathcal{P}$ can be seen in Figure 2.

2) Turbine Line Model: The second type of model we use is a line model $\mathcal{L} = \{\mathbf{l}_t, \mathbf{l}_n, \mathbf{l}_{b_1}, \mathbf{l}_{b_2}, \mathbf{l}_{b_3}\}$. These lines are defined by their end-points, which are taken from the previously defined point model $\mathcal{P}$

$$\mathcal{L} = \begin{cases} \mathbf{l}_t & = & \{\mathbf{p}_g, \mathbf{p}_t\} \\ \mathbf{l}_r & = & \{\mathbf{p}_t, \mathbf{p}_r\} \\ \mathbf{l}_{b_i} & = & \{\mathbf{p}_r, \mathbf{p}_{b_i}\} \end{cases} \tag{7}$$

The line $\mathbf{l}_t$ connects the base of the tower to the top of the tower. The line $\mathbf{l}_r$, connects the top of the tower and the centre of the blades. The lines $\mathbf{l}_{b_i}$ connect the centre of the blades with their tips. To instantiate an instance of $\mathcal{L}$ given a set of parameters θ, we use the set of functions Ψ defined in (2), applying them to each of the line end-points. This gives us a set of lines running along the centre of the main structural pieces of the wind turbine. An example of $\mathcal{L}$ can be seen in the connecting lines in Figure 2.

The reason for using a line model in combination with a point model is that during the inspection, there a numerous instances when only a small section of the wind turbine is in view. As these models are used to constrain our optimiser, if we were to rely solely on a $\mathcal{P}$, very few constraints would be added. By using $\mathcal{L}$ in combination with $\mathcal{P}$, as long as some of the turbine structure is in view of the drone, constraints can be established.

B. Turbine Projection Estimation using CNNs

To enable the non-linear least squares optimisation, we need to be able to establish correspondences between model points, and their 2D locations on images of the wind turbine. Due to the wide variation in the visual appearance of wind

turbines, accurately establishing correspondences using the raw image data is difficult. To address this we use a CNN to infer the projection of our skeletal model in the images captured by the drone, i.e. the projection of $\mathcal{P}$ and $\mathcal{L}$ into the camera coordinates, from the drone's current view of the turbine. This results in a set of projection images $\mathbf{P}_n = \left\{\mathbf{P}^{\mathcal{P}}_{n,g}, \mathbf{P}^{\mathcal{P}}_{n,t}, \mathbf{P}^{\mathcal{P}}_{n,r}, \mathbf{P}^{\mathcal{P}}_{n,b}, \mathbf{P}^{\mathcal{L}}_{n,t}, \mathbf{P}^{\mathcal{L}}_{n,r}, \mathbf{P}^{\mathcal{L}}_{n,b}\right\}$. Important to note are the superscripts referring to the type of model, the subscript n associating the images with a vertex in the pose graph (see Section III-C) and the second subscript referring to the type of structure within the model. The set of projection images are used to establish correspondences between the models, and locations in the images. Example outputs of the CNN can be seen in Figure 3. The network is trained using manually labelled images of wind turbines. Readers are referred to [5] for a full description of the CNN and training process.

C. Optimisation

The parameter estimation problem is solved using a non-linear least squares optimisation over a pose graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ and a set of turbine parameters θ. The graph vertices $\mathcal{V} = \{\mathbf{v}_n\}_{n=1}^{N}$, where N is the total number of vertices in the graph, each contain a full 6DoF pose $\mathbf{T}_n$, with the orientation represented by a quaternion $\mathbf{q}_n$ and a translation $\mathbf{t}_n$. The edges connect the vertices sequentially $\mathcal{E} = \{i, i+1\}_{i=1}^{N-1}$, mirroring the sequential addition of new vertices to the graph during inspection. The total set of parameters that we are optimising is then $G = \{\theta\} \bigcup \{\mathbf{q}_n, \mathbf{t}_n\}_{n=1}^{N}$. We define the cost function to be optimised as

$$E_{\text{cost}} = \lambda_{\mathcal{P}} E^{\mathcal{P}}_{\text{unary}} + \lambda_{\mathcal{L}} E^{\mathcal{L}}_{\text{unary}} + \lambda E_{\text{pairwise}} \quad . \tag{8}$$

The two unary terms measure the accuracy of the current parameter estimates based on the image data using points from $\mathcal{P}$ and points from $\mathcal{L}$. The pairwise term regularises the cost function by penalising poses that differ from pose measurements obtained by the GPS/IMU.

1) Unary Cost Function Term: Both unary terms E_{unary} in (8) are represented by the same function. We use the superscript to designate that one is computed using the points from $\mathcal{P}$ and one computed using the points from $\mathcal{L}$. We will use the notation $\mathcal{M}$ to equal either $\mathcal{P}$ or $\mathcal{L}$. In this section we present two different forms of $E^{\mathcal{M}}_{\text{unary}}$; one based on 2D point correspondences following the work in [5] $E^{\mathcal{M}}_{\text{unary (1)}}$, and a new method $E^{\mathcal{M}}_{\text{unary (2)}}$ based on direct evaluation of the pixel values in the projection images $\mathbf{P}_n$. Necessary for both these formulations of the unary term is the following projection function

$$\mathbf{a}_{n,m} = \Pi\left(\mathbf{v}_n, \mathbf{p}_m\right) = \mathbf{K}[\mathbf{R}\left(\mathbf{q}_n\right)|\mathbf{t}_n] \;\; \mathbf{p}_m \quad . \tag{9}$$

Typical in monocular tracking applications, this function allows us to transform a point $\mathbf{p}_m$ from the world frame into camera image coordinates $\mathbf{a}_{n,m}$ using the transformation $\mathbf{q}_n$ and $\mathbf{t}_n$ held in a vertex $\mathbf{v}_n$, and $\mathbf{K}$ which represents the camera intrinsics. The first version of the unary term we present is based on 2D point correspondences and is formulated as

$$E^{\mathcal{M}}_{\text{unary (1)}} = \sum_{\mathbf{v}_n \in \mathcal{V}} \sum_{\hat{\mathbf{p}}_m \in \hat{\mathcal{M}}} \left\| \Pi\left(\mathbf{v}_n, \psi_m(\hat{\mathbf{p}}_m, \theta)\right) - \mathbf{a}'_{n,m} \right\|^2 \quad . \tag{10}$$

The value $\mathbf{a}'_{n,m}$ is the 2D correspondence to the model point $\hat{\mathbf{p}}$, on the projection image $\mathbf{P}^{\mathcal{M}}_{n,m}$. This term transforms the model point into the world frame using the functions Ψ, and then projects it into the image using (9). One important process is establishing these correspondences prior to optimising. Depending on the type of model, correspondences are found in different ways.

The first uses the point model $\mathcal{P}$. We take each of the current estimates of the poses in the graph $\mathbf{v}^*_n \in \mathcal{V}^*$ and the current estimate of the parameters θ^* and project each of the model points $\hat{\mathbf{p}}_m$ into the image.

$$\mathbf{a}^*_{n,m} = \Pi\left(\mathbf{v}^*_n, \psi_m(\hat{\mathbf{p}}_m, \theta^*)\right) \quad . \tag{11}$$

We then search the image pixels in $\mathbf{P}^{\mathcal{P}}_{n,m}$ around $\mathbf{a}^*_{n,m}$ and find the pixel with the largest value. This pixel, if the value is above a threshold is selected as $\mathbf{a}'_{n,m}$. This process is illustrated in Figure 4.

The second method using the line model $\mathcal{L}$ is somewhat more involved. The reason for using the line model in the optimiser is that as the drone is often close to the turbine during the inspection, the points in $\mathcal{P}$ are not always visible. Using $\mathcal{L}$ allows us to establish correspondences in-between the line end points. To do this we split each of the lines in $\mathcal{L}$ into points $\hat{\mathbf{p}}_m$, where $m \in \{t, r, b\}$ and use these to establish correspondences.

Similar to the points in $\mathcal{P}$, for each split point $\hat{\mathbf{p}}_m$, we find the 2D projection $\mathbf{a}^*_{m,n}$ using (11). Using the projected location, we then perform a perpendicular line search from

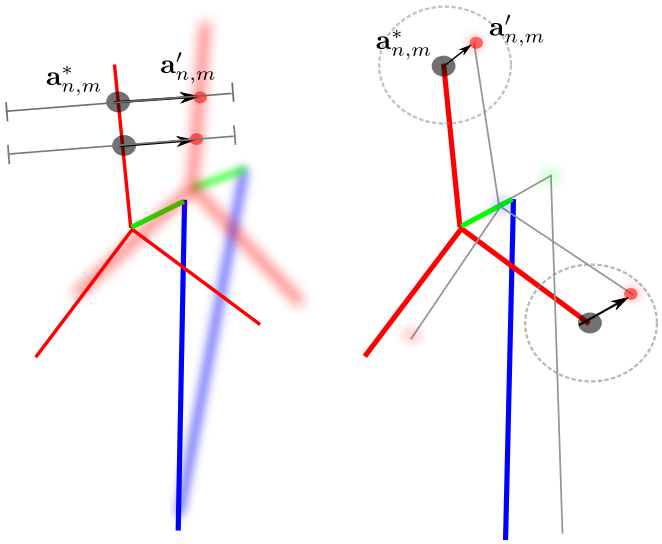

Fig. 4. Figure showing how correspondences are established. Model projections $\mathbf{P}$ inferred by the CNN are shown as blurred points and lines and the model projection of $\mathcal{L}$ and $\mathcal{P}$ using the current pose estimate is shown in bold colours. Left) Line-based correspondences are found using a perpendicular line search. Right) Point based correspondences are found by searching within a circular window.

$\mathbf{a}^*_{n,m}$. The pixel along the line in $\mathbf{P}^{\mathcal{L}}_{n,m}$ with the highest value above a threshold is then chosen as $\mathbf{a}'_{n,m}$. This process is illustrated in Figure 4. For a more detail on the process of establishing correspondences, please see [5].

The second formulation of the unary cost function $E^{\mathcal{M}}_{\text{unary (2)}}$ is based on directly sampling the pixel values in $\mathbf{P}^{\mathcal{M}}_{n,m}$ using the projection of the model points into the image and interpolating the values during optimisation. The output of the CNN can be viewed as a heat map, with pixel values ranging between $[0, 1]$. The higher the pixel value, the great the expectation that the corresponding turbine feature lies at that location. We take this as a probability, and optimising the cost function becomes a process of finding the parameters that project the point model to the locations on the image with the highest probability, corresponding to the minimum of the sum of the negative log over all points, i.e.

$$E^{\mathcal{M}}_{\text{unary (2)}} = \sum_{\mathbf{v}_n \in \mathcal{V}} \sum_{\hat{\mathbf{p}}_m \in \hat{\mathcal{M}}} -\log\left(\Gamma\left(\mathbf{P}^{\mathcal{M}}_{n,m}, \Pi\left(\mathbf{v}_n, \Psi_m(\hat{\mathbf{p}}_m, \theta)\right)\right)\right) \quad . \tag{12}$$

The function Γ uses bicubic interpolation to evaluate the pixel value under the projected point using Cubic Hermite Splines. Importantly, this type of interpolation is differentiable, allowing us to integrate the function into the non-linear optimiser.

In this application, there are a number of benefits of using the direct image interpolation method $E^{\mathcal{M}}_{\text{unary (2)}}$ over the 2D correspondence method $E^{\mathcal{M}}_{\text{unary (1)}}$. It means we don't need to establish correspondences prior to optimisation, reducing the complexity of the process. More importantly, the interpolation method allows the model point projections to move freely in image space during optimisation. This is especially important for the points from $\mathcal{L}$. If we use the 2D point correspondences as defined in $E^{\mathcal{M}}_{\text{unary (1)}}$, the optimiser is discouraged from using update steps that will move line points in the same direction as the corresponding line in the projection image $\mathbf{P}$. This is unwanted, as the cost incurred by a line point should be equal anywhere along the length of the corresponding line in $\mathbf{P}$.

2) Pairwise Cost Function Term: The second term of the cost function E_{pairwise} is used to constrain the pose parameters during optimisation such that they don't differ too far from the original estimates of the drone's pose measured using GPS/IMU. To enable this we compute the relative poses $\mathbf{T}_{i,j} \; \forall \; \{i, j\} \in \mathcal{E}$ in the following way

$$\mathbf{T}_{i,j} = \begin{bmatrix} \mathbf{t}_{i,j} \\ \mathbf{q}_{i,j} \end{bmatrix} = \begin{bmatrix} \mathbf{R}\left(\mathbf{q}_i\right)^T \left(\mathbf{t}_j - \mathbf{t}_i\right) \\ \mathbf{q}_i^{-1} * \mathbf{q}_j \end{bmatrix} \quad , \tag{13}$$

where $\mathbf{R}\left(\mathbf{q}_i\right)$ is the rotation matrix formed from the quaternion rotation $\mathbf{q}_i$. Using relative measurements as constraints rather than absolute measurements is beneficial in that it allows the optimised poses to drift over time – correcting for drift in the GPS/IMU – while harshly penalising abrupt differences between successive key frames. The full pairwise term is then defined as

$$E_{\text{pairwise}} = \sum_{i,j \in \mathcal{E}} \mathbf{C} \begin{bmatrix} \hat{\mathbf{t}}_{i,j} - \mathbf{t}_{i,j} \\ 2 \times \text{Vec}\left(\hat{\mathbf{q}}_{i,j} * \mathbf{q}_{i,j}^{-1}\right) \end{bmatrix} \quad , \tag{14}$$

where 'Vec' corresponds to the vector component of the quaternion. The matrix $\mathbf{C}$ is an information matrix, which encodes the certainty about the relative pose measurements obtained by the GPS/IMU. Due to limitations on our system, we don't have access to this data during the flight. Instead we fix this matrix using a set of reasonable values. Further details can be found in [5].

To optimise (8), we use the Levenberg-Marquardt [13] non-linear least squares method. This gives us our fitted pose and turbine parameter values. We make use of the Ceres Solver [14] software library to perform the optimisation process.

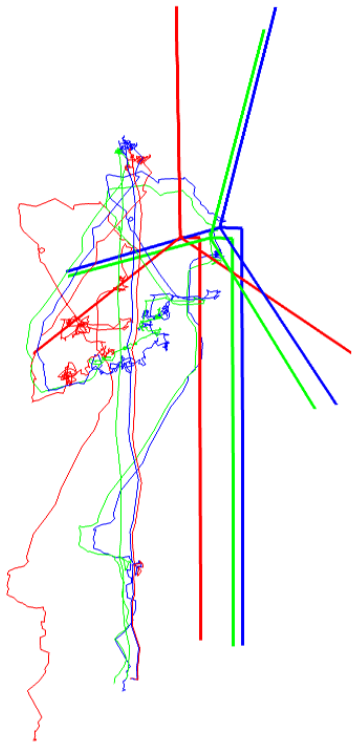

Fig. 5. Visual example of the combined pose and model optimisation process. The green turbine corresponds to the ground truth $\mathcal{P}$, the red to the initial configuration of $\mathcal{P}$ and the blue to the optimised configuration of $\mathcal{P}$. Similarly, the green path corresponds to the ground truth positions of the drone, the red to the initial positions and the blue to the optimised positions.

IV. EVALUATION

To evaluate our work we present two types of experiments. The first in Section IV-A relies on simulated data, allowing us to empirically evaluate the performance of our method against a known ground truth. The second set in Section IV-B applies our work to real wind turbine inspection data. As this data does not have a known ground truth, we only provide a qualitative evaluation of the results. The following

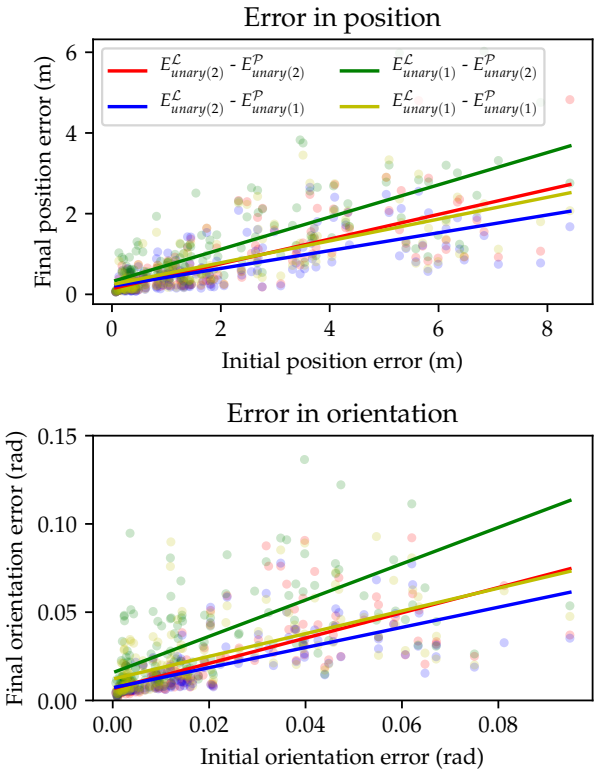

Fig. 6. Plots comparing the performance of the different types of unary cost on location accuracy (top) and orientation accuracy (bottom). The plot legend labels correspond to the cost term used for the points in $\mathcal{L}$ and $\mathcal{P}$. The solid lines show the LS fit line for each configuration.

evaluations were performed offline on a laptop with an intel i7 processor and a 16gb memory. The framework is able to process a new keyframe and optimise at ∼5hz. We also have an implementation that runs on a Nvidia Jetson Tx2, which is able to process frames and optimise at ∼1hz.

A. Simulated Data

To provide a qualitative evaluation of our method we applied it to a set of simulated inspection data. To generate the synthetic data, we extracted pose graphs and turbine parameter sets from a number of real inspection flights which we use as pseudo ground truths. This provides us with a representative set of data for evaluation. For each of these ground truths we applied varying levels of noise to the data. For the pose graphs, we sequentially added noise to the position and orientations, allowing us to simulate the sort of drift one might expect from using GPS/IMU readings during a flight. For the turbine parameters we added error from a Gaussian distribution to each of the values. Due to the different scales for the parameters in θ, the σ values for each parameter were different. Using this data we evaluated a number of different aspects of our method. An example of the output can be seen in Figure 5.

1) Unary cost function comparison: The first aspect we evaluated was the difference in performance of the two unary cost terms we defined in Section III-C. These are the one based on 2D point correspondences $E^{\mathcal{M}}_{\text{unary (1)}}$ and the one based on direct image interpolation $E^{\mathcal{M}}_{\text{unary (2)}}$. As we have two different sets of model points that these can be applied to – the points from $\mathcal{P}$ and the points from $\mathcal{L}$ – we compared a total of 4 different configurations. We applied each of these configurations to our set of synthetic data and recorded the mean error in position and orientation prior to optimisation, and the same values after optimisation. The results of the experiments can be seen in Figure 6. As the plots show, the method that uses $E^{\mathcal{M}}_{\text{unary (2)}}$ for $\mathcal{L}$ and the $E^{\mathcal{M}}_{\text{unary (1)}}$ for $\mathcal{P}$ performs best for both position and orientation. The method that performs the worse is when this configuration is reversed. This performance is expected due to the factors highlighted in Section III-C. As the points in $\mathcal{P}$ correspond to actual 2D correspondences in the image, using the point correspondence cost $E^{\mathcal{M}}_{\text{unary (1)}}$ makes sense. As the points in $\mathcal{L}$ should contribute a similar amount to the cost at any location along the line of the corresponding turbine feature, the more relaxed $E^{\mathcal{M}}_{\text{unary (2)}}$ function is appropriate. Given these results, for the remainder of the experiments we use $E^{\mathcal{M}}_{\text{unary (2)}}$ for $\mathcal{L}$ and $E^{\mathcal{M}}_{\text{unary (1)}}$ for $\mathcal{P}$.

2) Combined Optimisation Comparison: The second aspect of our method we evaluated was the benefit of simultaneously optimising both the pose and turbine parameters over optimising just the pose. To do this we applied our method to the set of synthetic data twice; once where both sets of parameters were optimised, once where only the pose was optimised. We then evaluated the error of both methods. The results of this experiment can be seen in Figure 7. As these figures show, when we are optimising both sets of parameters, the method is significantly improved over optimising the pose alone. The reason for this is that errors in the turbine parameters will propagate into errors in the pose of the drone, if the model parameters are fixed during optimisation. The most obvious instance of this is when the value ϕ – which sets the rotation of the blades – is wrong. During optimisation, this will propagate into error in the roll as the drone seeks to align the incorrect model with the image data. We can see this in Figure 9.

3) Recovering the Turbine Parameters: The final set of synthetic evaluations we performed was to establish the ability of the method to optimise the parameters of the turbine model. To do this we applied the set of functions Ψ to the ground truth parameters, the initial parameters (those with noise added) and the optimised parameters. We then computed the mean distance error across all the points in $\mathcal{P}$ for the initial model and the optimised model and plotted them against each other. The results of this can be seen in Figure 8. We can see that as expected, when there is more error in the initial turbine parameters, the optimiser is less able to recover to the true pose. We do see however, that the method is able to recover around half the error introduced into the initial parameters.

B. Real Data

To give an example of the performance of our method we applied it to some recordings of real inspection flights.

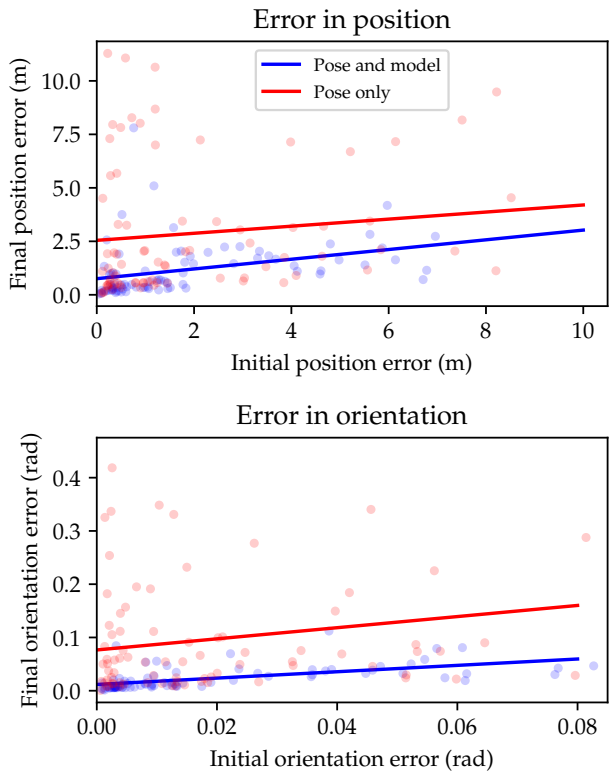

Fig. 7. Plot comparing the performance of our combined method over the method that optimises only the pose. Top) Position error, Bottom) Orientation Error.

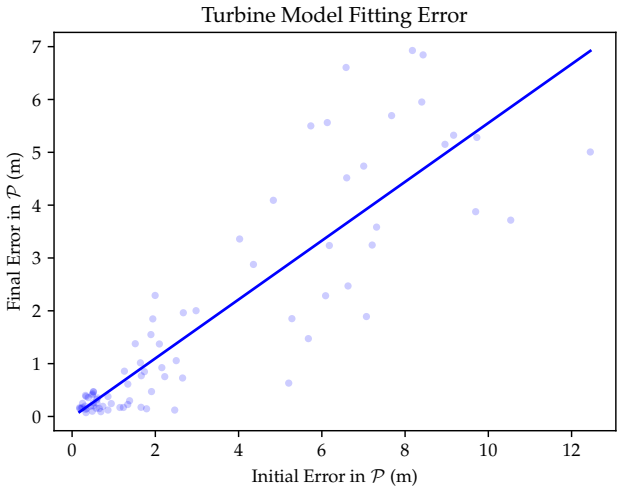

Fig. 8. Plot showing the performance of our method for fitting the turbine model parameters. Plot shows initial error in meters against error after optimisation.

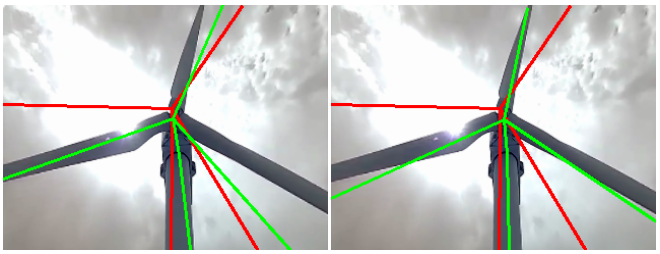

Fig. 9. Image comparing pose only optimisation with joint pose and model optimisation on real data. The red turbine projection is the initial estimate. The green projection is the optimised estimate. Left) Output from pose only optimisation. Right) Output from combined optimisation.

For this data, the initial pose estimates are measured using GPS/IMU readings and the turbine model parameters were initialised by eye. In Figure 9, we give an example of the benefit of optimising both model and pose parameters simultaneously. As we can see, when there is error in blade rotation value ϕ, if we are not able to correct for that error during the flight, the optimiser will be encouraged to correct for it by incorrectly adding roll to the orientation.

In Figure 10, we show a series of images showing the reprojection of the turbine model. We can see that our method is able to correct for a large proportion of the error in the initial estimate of the turbine and pose.

V. CONCLUSIONS

In this work we have presented a method for the simultaneous optimisation of a drone's pose as well the parameters of a wind turbine model during an inspection flight. We present a set of functions that allow us to integrate the model parameter fitting into a non-linear least squares optimiser. In addition we also present a new cost function term which is better suited to the image data than the traditional corresponding points method. In our evaluations, we have compared the two types of unary cost terms, shown the benefit of the simultaneous optimisation using both simulated and real data, and measured how effective our method is at correcting errors in the model parameters.

There are a number of avenues open for future work. One of these is the incorporation of extra sensor measurements into the optimisation. This could take the form of depth data from a LiDAR or from stereo cameras. We also aim to properly integrate the GPS/IMU measurements using a visual inertial odometry method.

ACKNOWLEDGEMENTS

The authors acknowledge the support of Innovate UK (project number 104067). The work described herein is the subject of UK patent applications GB1815864.2 and GB1902475.1.

REFERENCES

[1] J. Ribrant and L. M. Bertling, "Survey of failures in wind power systems with focus on Swedish wind power plants during 1997-2005," *IEEE Transactions on Energy Conversion*, vol. 22, no. 1, pp. 167–173, mar 2007.

[2] F. P. García Márquez, A. M. Tobias, J. M. Pinar Pérez, and M. Papaelias, "Condition monitoring of wind turbines: Techniques and methods," *Renewable Energy*, vol. 46, pp. 169–178, oct 2012.

[3] L. Wang and Z. Zhang, "Automatic Detection of Wind Turbine Blade Surface Cracks Based on UAV-Taken Images," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 9, pp. 7293–7309, sep 2017.

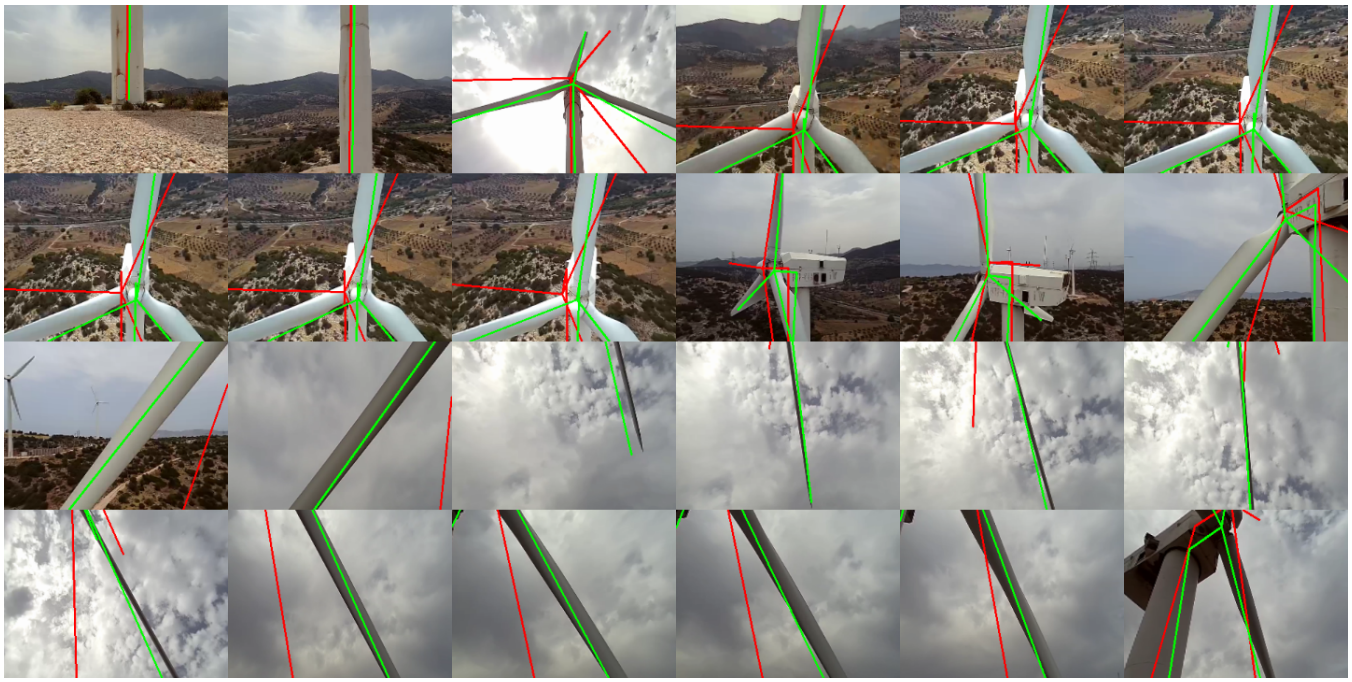

Fig. 10. Frames taken from a real inspection. The red turbine is the reprojection using the initial pose parameters and turbine model parameters. The green turbine is the reprojection from the optimised pose parameters and turbine model parameters.

[4] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, mar 2015.

[5] O. Moolan-Feroze, K. Karachalios, D. N. Nikolaidis, and A. Calway, "Improving drone localisation around wind turbines using monocular model-based tracking," feb 2019.

[6] M. Stokkeland, K. Klausen, and T. A. Johansen, "Autonomous visual navigation of Unmanned Aerial Vehicle for wind turbine inspection," in *2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015*. IEEE, jun 2015, pp. 998–1007.

[7] B. E. Schäfer, D. Picchi, T. Engelhardt, and D. Abel, "Multicopter unmanned aerial vehicle for automated inspection of wind turbines," in *24th Mediterranean Conference on Control and Automation, MED 2016*. IEEE, jun 2016, pp. 244–249.

[8] K. Melbouci, S. N. Collette, V. Gay-Bellile, O. Ait-Aider, and M. Dhome, "Model based RGBD SLAM," in *Proceedings - International Conference on Image Processing, ICIP*, vol. 2016-Augus. IEEE, sep 2016, pp. 2618–2622.

[9] A. Loesch, S. Bourgeois, V. Gay-Bellile, O. Gomez, and M. Dhome, "Localization of 3D objects using model-constrained SLAM," *Machine Vision and Applications*, vol. 29, no. 7, pp. 1041–1068, oct 2018.

[10] K. Pauwels, L. Rubio, and E. Ros, "Real-time model-based articulated object pose detection and tracking with variable rigidity constraints," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2014, pp. 3994–4001.

[11] F. Michel, S. Gumhold, E. Brachmann, M. Y. Yang, C. Rother, and A. Krull, "Pose Estimation of Kinematic Chain Instances via Object Coordinate Regression," in *Procedings of the British Machine Vision Conference 2015*. British Machine Vision Association, 2015, pp. 181.1–181.11.

[12] D. Katz, M. Kazemi, J. Andrew Bagnell, and A. Stentz, "Interactive segmentation, tracking, and kinematic modeling of unknown 3D articulated objects," in *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, may 2013, pp. 5003–5010.

[13] D. W. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," vol. 11, no. 2, pp. 431–441, jun 2005.

[14] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.